UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/660,353	09/11/2003	P. Anders I. Bertelrud	2095.001100/P3126	5128

62293 7590 05/26/2009
WILLIAMS, MORGAN & AMERSON, P.C.
10333 RICHMOND AVE.
SUITE 1100
HOUSTON, TX 77042

EXAMINER	
KISS, ERIC B	
ART UNIT	PAPER NUMBER
2192	
MAIL DATE	DELIVERY MODE
05/26/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/660,353	BERTELRUD ET AL.	
	Examiner	Art Unit	
	ERIC B. KISS	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE _3_ MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on _06 February 2009_.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) _1-4,6-13,15-21 and 23-37_ is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) _1-4,6-13,15-21 and 23-37_ is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application

6) ☐ Other: _____.

DETAILED ACTION

1. The reply filed February 6, 2009, has been received and entered. Claims 1-4, 6-13, 15-21, 23-37 are pending.

Response to Amendment

2. The rejection of claim 16 under 35 U.S.C. § 101 is withdrawn in view of applicants' amendment.

Response to Arguments

3. Applicant's arguments with respect to the rejections under §§ 102(b) and 103(a) have been fully considered but they are not persuasive.

Contrary to the applicants' suggestions, the compiler of McKeeman is not forced to recompile whole files in response to changes in source code, but rather performs incremental recompiling, so that if only one line is changed in an edit session, then only that line and lines related to it need to be recompiled if no other code is affected. See, e.g, McKeeman, Abstract at lines 4-8 and 16-18. See also *Id.* at col. 16, line 60, through col. 17, line 6. Because the previous compilation results for portions of the file that have not been modified are reused in subsequent compilation, the previous compilation may be reasonably interpreted as initiating (and completing) compilation prior to the subsequent request for recompilation. As noted in the previous Office action, McKeeman's approach is iterative, and the compilation from one iteration precedes a request to compile in a later iteration. The unmodified portion of the modified files is compiled before, and reused during, a subsequent request to recompile.

Claim Rejections - 35 USC § 102

4. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

5. Claims 1-4, 6-13, 15-21, 23-29, and 31-37are rejected under 35 U.S.C. 102(b) as being anticipated by U.S. Patent No. 5,193,191 (McKeeman et al.).

As per claim 1, *McKeeman et al.* discloses:

initiating compilation of a file in a processor-based system in advance of a request from a user to compile the file (see, e.g., col. 11, lines 44-61 (recompilation uses previously compiled code);

detecting the user request to compile the file (see, e.g., *Id.* (recompilation is started); and

indicating a status of the compilation of the file in response to detecting the user request (see, e.g., col. 5, lines 23-34 (errors are detected and reported));

wherein initiating compilation of the file comprises compiling the file in response to determining that the file has been modified (see, e.g., col. 5, lines 21-23).

As per claim 2, *McKeeman et al.* further discloses initiating compilation of the file comprising compiling the file including one or more code segments to produce an object code file (see, e.g., col. 5, lines 30-34).

As per claim 3, *McKeeman et al.* further discloses compiling the file comprising compiling one or more code segments in the file to produce an object code file, and further comprising linking the object code file to produce an executable file (see, e.g., col. 5, lines 37-46).

As per claim 4, *McKeeman et al.* further discloses indicating the status of the compilation of the file comprising at least one of indicating that the compilation was successful and indicating that the compilation was unsuccessful (see, e.g., col. 5, lines 23-34).

As per claim 6, *McKeeman et al.* further discloses determining that the file has been modified comprising determining that the modified file has been saved to a storage unit (see, e.g., col. 5, lines 18-23).

As per claim 7, *McKeeman et al.* further discloses the file including one or more code segments, wherein initiating compilation of the file in response to determining that the file has been modified comprises:

identifying the modified file in a work queue (see, e.g., col. 11, lines 44-61); and

initiating the compilation of the file based on the modified file being identified in the work queue (see, e.g., col. 11, lines 44-61).

As per claim 8, *McKeeman et al.* further discloses indicating the status of the compilation of the file comprising generating one or more files associated with the compilation of the file, storing the one or more generated files in a temporary location, and transferring the one or more files from the temporary location to a different location in response to detecting the user request (see, e.g., col. 5, lines 30-34).

As per claim 9, *McKeeman et al.* discloses an article comprising one or more machine-readable storage media containing instructions (see, e.g., col. 8, lines 6-39) that when executed enable a processor to:

initiate compiling of a file including one or more code segments (see, e.g., col. 11, lines 44-61 (recompilation uses previously compiled code);

detect a user request to compile the file (see, e.g., *Id.* (recompilation is started)); and

provide a result associated with the compiling in response to detecting the user request (see, e.g., col. 5, lines 23-34 (errors are detected and reported));

wherein the instructions when executed enable the processor to initiate compiling of the file based on determining that the file was modified (see, e.g., col. 5, lines 21-23).

As per claim 10, *McKeeman et al.* further discloses the instructions when executed enable the processor to display a message to a user indicating that one or more errors were detected during the compiling (see, e.g., col. 5, lines 23-34 (errors are detected and reported)).

As per claim 11, *McKeeman et al.* further discloses the instructions when executed enable the processor to indicate to a user that the compiling was successful (see, e.g., col. 5, lines 23-34 (errors are detected and reported; if no errors are detected, then object code is produced as input to the linker)).

As per claim 12, *McKeeman et al.* further discloses the instructions when executed enable the processor to generate a file containing object code based on compiling the file and to store the object code file in a temporary location (see, e.g., col. 5, lines 30-34).

As per claim 13, *McKeeman et al.* further discloses the instructions when executed enable the processor to move the object code file from the temporary location into a product location based on determining that the compiling of the file was successful and in response to detecting the user request (see, e.g., col. 5, lines 23-34 (errors are detected and reported; if no errors are detected, then object code is produced as input to the linker)).

As per claim 15, *McKeeman et al.* further discloses the the instructions that when executed enable the processor to initiate compiling of the file based on determining that the file was modified comprise instructions that when executed enable the processor to indicate in a work queue that the file has been modified and to initiate compiling of the file in response to detecting the indication (see, e.g., col. 11, lines 44-61).

As per claim 16, *McKeeman et al.* discloses an apparatus (see, e.g., col. 8, lines 6-39), comprising:

means for initiating compilation of a file in a processor-based system in advance of a request from a user (see, e.g., col. 11, lines 44-61 (recompilation uses previously compiled code));

means for detecting the user request to compile the file (see, e.g., *Id.* (recompilation is started)); and

means for indicating a status of the compilation of the file in response to detecting the user request (see, e.g., col. 5, lines 23-34 (errors are detected and reported));

wherein the means for initiating compilation initiate compiling the file based on determining that the file was modified (see, e.g., col. 5, lines 18-23; col. 13, lines 42-54).

McKeeman et al. further discloses a machine-readable storage media containing instructions for implementing the recited functionality (see, e.g., col. 8, lines 6-39).

As per claim 17, *McKeeman et al.* discloses an apparatus (see, e.g., col. 8, lines 6-39), comprising:

a storage unit having a file stored therein (see, e.g., col. 8, lines 6-39); and a control unit communicatively coupled to the storage unit (see, e.g., col. 8, lines 6-39), the control unit adapted to:

initiate compilation of the file in advance of a request from a user to compile the file (see, e.g., col. 11, lines 44-61 (recompilation uses previously compiled code));

detect the user request to compile the file (see, e.g., *Id.* (recompilation is started)); and

indicate a status of the compilation of the file in response to detecting the user request (see, e.g., col. 5, lines 23-34 (errors are detected and reported));

wherein the control unit is adapted to compile the file in response to determining that the file has been modified (see, e.g., col. 5, lines 18-23; col. 13, lines 42-54).

As per claim 18, *McKeeman et al.* discloses the control unit is adapted to compile a file including one or more code segments to produce an object code file (see, e.g., col. 5, lines 30-34).

As per claim 19, *McKeeman et al.* discloses the control unit is adapted to link the object code file to produce an executable file (see, e.g., col. 5, lines 37-46).

As per claim 20, *McKeeman et al.* discloses the control unit is adapted to store the executable file in a temporary location and to transfer the executable file from the temporary location to a different location based on detecting the user request (see, e.g., col. 5, lines 30-34).

As per claim 21, *McKeeman et al.* discloses the control unit is adapted to at least one of indicate that the compilation was successful and indicate that the compilation was unsuccessful (see, e.g., col. 5, lines 23-34).

As per claim 23, *McKeeman et al.* discloses the control adaptation to compile the file in response to determining that the file has been modified comprises:

an adaptation to identify the modified file in a work queue (see, e.g., col. 11, lines 44-61); and

an adaptation to initiate the processing of the file based on the modified file being identified in the work queue (see, e.g., col. 11, lines 44-61).

As per claim 24, *McKeeman et al.* discloses:

identifying one or more source files that have been modified in a processor-based system (see, e.g., col. 5, lines 18-23; col. 13, lines 42-54);

initiating processing of at least a portion of the modified source files before receiving a request to process the modified files (see, e.g., col. 11, lines 44-61);

receiving the request to process at least one of the modified files (see, e.g., col. 11, lines 44-61 (recompilation uses previously compiled code)); and

providing a status associated with the processing of the file in response to receiving the request (see, e.g., col. 5, lines 23-34 (errors are detected and reported)).

As per claim 25, *McKeeman et al.* further discloses the processor-based system is adapted to execute an integrated development environment module (see, e.g., col. 3, lines 53-56), wherein identifying the one or more files comprises the integrated development environment module placing the one or more of the source files that have been modified in a queue (see, e.g., col. 11, lines 44-61).

As per claim 26, *McKeeman et al.* further discloses placing the one or more of the source files in the queue comprises placing at least one source file in the queue in response to a user saving the source file to a storage unit (see, e.g., col. 5, lines 18-23).

As per claim 27, *McKeeman et al.* further discloses placing the one or more of the source files in the queue comprises placing at least a portion of one source file in the queue in response to a user saving the source file to a storage unit using an editor and then exiting from the editor (see, e.g., col. 17, lines 11-36).

As per claim 28, *McKeeman et al.* further discloses placing the one or more of the source files in the queue comprises placing at least one source file in the queue in response to determining that a user desires to compile at least a portion of the source file as the source file is being edited (see, e.g., col. 5, lines 15-17; col. 11, lines 44-61).

As per claim 29, *McKeeman et al.* further discloses placing the one or more of the source files in the queue comprises placing at least one source file in the queue in response to determining that the source file includes at least one marker identifying a section of the source file that should be compiled, and wherein initiating processing of at least the portion of the one or more modified files comprises compiling the identified section of the source file (see, e.g., col. 11, lines 44-61).

As per claim 31, *McKeeman et al.* further discloses initiating the processing comprises initiating a build process to produce a software application and wherein receiving the request comprises receiving the request to building the software application (see, e.g., col. 5, lines 15-17).

As per claim 32, *McKeeman et al.* further discloses initiating the build process comprises performing compiling the modified source files to produce object code files and linking the object code files to produce executable files (see, e.g., col. 5, lines 37-46).

As per claim 33, *McKeeman et al.* further discloses placing the one or more of the source files in the queue comprising placing at least one source file in the queue in response to determining that the source file includes at least two markers identifying a section of the source file that should be compiled, wherein the first marker defines the beginning of the portion of the source file and the second marker defines the end of the portion, and wherein initiating processing of at least the portion of the one or more modified files comprises compiling the identified section of the source file (see, e.g., col. 16, line 60, through col. 17, line 6 (the compiler of *McKeeman* can identify changed portions (having a defined beginning and end) within a source code file and recompile only those changed portions)).

As per claim 34, *McKeeman et al.* further discloses suppressing at least one of an error and warning that is detected while compiling the modified source files (see, e.g., col. 5, lines 23-34 (errors are detected and reported)).

As per claim 35, *McKeeman et al.* further discloses the object code files and the executable files are moved to a different storage location in response to detecting the request and in response to detecting no error or warning (see, e.g., col. 5, lines 23-34 (errors are detected and reported; if no errors are detected, then object code is produced as input to the linker)).

As per claim 36, *McKeeman et al.* further discloses identifying one or more source files comprises identifying the one or more source files based on a directed acyclic graph (see, e.g., col. 16, 54-63).

As per claim 37, *McKeeman et al.* further discloses the directed acyclic graph includes a list of dependent files, wherein identifying one or more source files comprises identifying at least one modified source file and another source file that is dependent on the modified source file using the directed acyclic graph (see, e.g., col. 16, 54-63).

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claim 30 is rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 5,193,191 (McKeeman et al.) in view of U.S. Pat. App. Pub. No. 2005/0108682 (Piehler et al.).

As per claim 30, *McKeeman et al.* discloses such a method (see the rejection of claims 24 and 25 under 35 U.S.C. § 102(b)) but fails to expressly disclose initiating the processing of the modified source files comprises causing a background thread to awaken in response to placing the one or more of the source files in the queue, where the background thread thereafter initiates processing of the source files. However, *Piehler et al.* teaches such use of a background thread to enqueue a task for itself to complete the rest of the compilation in a background thread as part of a response to a new file being added to a project or an existing file being modified (*Piehler et al.* at paragraphs [0170] through [0180]. Therefore, it would have been obvious to include such background thread use as per the teachings of *Piehler et al.* One would be motivated to do so to gain the advantage of providing immediate feedback to the user without any detectable visible delay in typing responsiveness while the compiler finishes processing the change (*Poehler et al.* at paragraph [0174]).

Conclusion

8. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Ed Robinson, et al., Upgrading Microsoft Visual Basic 6.0 to Microsoft Visual Basic .NET, January 5, 2002, Microsoft Press, pp. 1-18, discusses the background compiler introduced in the VISUAL BASIC .NET development environment prior to applicants' invention. Specifically, the Microsoft background compiler is disclosed as continually working in the background and resulting in compilation errors being flagged in code in real time with blue squiggle underlines. See p. 16. Further, the background compiler is disclosed as parsing a typed statement as soon as the cursor is moved to a different line within the editor and adding detected compile errors to a Task List (and removed from the task list automatically upon correction of the error). See p. 18.

Matthew Gertz, "Advanced Basics; Scaling Up: The Very Busy Background Compiler," Microsoft Corp., MSDN Magazine, June 2005, 4 pages, discusses in more detail the implementation of the VB.NET background compiler and the changes in its design from the VISUAL BASIC .NET 2002 development environment to the VISUAL BASIC .NET 2005 development environment.

U.S. 2005/0114771 (Piehler et al.) (claiming the benefit to provisional application 60/449,984, filed February 26, 2003) discloses an editor and compiler integrated with a retokenizer that allows incremental compilation of code in the background as the user types it in the editor (paragraphs [0165] through [0176]).

9. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

10. Any inquiry concerning this communication or earlier communications from the Examiner should be directed to Eric B. Kiss whose telephone number is (571) 272-3699. The Examiner can normally be reached on Tue. - Fri., 7:00 am - 4:30 pm. The Examiner can also be reached on alternate Mondays.

If attempts to reach the Examiner by telephone are unsuccessful, the Examiner's supervisor, Tuan Dam, can be reached on (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is (571) 273-8300.

/Eric B. Kiss/
Eric B. Kiss
Primary Examiner, Art Unit 2192